01-23-06

HEWLETT-PACKARD COMPANY
Intellectual Property Administration
P.O. Box 272400
Fort Collins, Colorado 80527-2400

PATENT APPLICATION

ATTORNEY DOCKET NO. 200308263-1

IN THE
UNITED STATES PATENT AND TRADEMARK OFFICE

Inventor(s):	Jeffrey C. Mogul	Confirmation No.:	3408
Application No.:	09/825,661	Examiner:	D.A.C. Perez
Filing Date:	April 30, 2001	Group Art Unit:	2154

Title: REDUCTION OF NETWORK RETRIEVAL LATENCY USING CACHE AND DIGEST

Mail Stop Appeal Brief-Patents
Commissioner For Patents
PO Box 1450
Alexandria, VA 22313-1450

TRANSMITTAL OF APPEAL BRIEF

Transmitted herewith is the Appeal Brief in this application with respect to the Notice of Appeal filed on November 21, 2005.

The fee for filing this Appeal Brief is (37 CFR 1.17(c)) $500.00.

(complete (a) or (b) as applicable)

The proceedings herein are for a patent application and the provisions of 37 CFR 1.136(a) apply.

☐ (a) Applicant petitions for an extension of time under 37 CFR 1.136 (fees: 37 CFR 1.17(a)-(d)) for the total number of months checked below:

☐ 1st Month $120	☐ 2nd Month $450	☐ 3rd Month $1020	☐ 4th Month $1590

☐ The extension fee has already been filed in this application.

☒ (b) Applicant believes that no extension of time is required. However, this conditional petition is being made to provide for the possibility that applicant has inadvertently overlooked the need for a petition and fee for extension of time.

Please charge to Deposit Account 08-2025 the sum of $ 500. At any time during the pendency of this application, please charge any fees required or credit any over payment to Deposit Account 08-2025 pursuant to 37 CFR 1.25. Additionally please charge any fees to Deposit Account 08-2025 under 37 CFR 1.16 through 1.21 inclusive, and any other sections in Title 37 of the Code of Federal Regulations that may regulate fees. A duplicate copy of this sheet is enclosed.

☒ I hereby certify that this correspondence is being deposited with the United States Postal Service as Express Mail, Label No. EV 568257431US addressed to: Commissioner for Patents, Alexandria, VA 22313-1450
Date of Deposit: January 20, 2006

OR

☐ I hereby certify that this paper is being transmitted to the Patent and Trademark Office facsimile number (571)273-8300.
Date of facsimile:
Typed Name: Joy H. Perigo
Signature: Joy H. Perigo

Respectfully submitted,
Jeffrey C. Mogul
By
Jody C. Bishop
Attorney/Agent for Applicant(s)
Reg No. : 44,034
Date : January 20, 2006
Telephone : (214) 855-8007

HEWLETT-PACKARD COMPANY
Intellectual Property Administration
P.O. Box 272400
Fort Collins, Colorado 80527-2400

Docket No.: 200308263-1
(PATENT)

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of:
Jeffrey C. Mogul

Application No.: 09/825,661 | Confirmation No.: 3408

Filed: April 3, 2001 | Art Unit: 2154

For: REDUCTION OF NETWORK RETRIEVAL LATENCY USING CACHE AND DIGEST | Examiner: D. A. C. Perez

APPEAL BRIEF

MS Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

As required under § 41.37(a), this brief is filed within two months of the Notice of Appeal filed in this case on November 21, 2005, and is in furtherance of said Notice of Appeal.

The fees required under § 41.20(b)(2) are dealt with in the accompanying TRANSMITTAL OF APPEAL BRIEF.

This brief contains items under the following headings as required by 37 C.F.R. § 41.37 and M.P.E.P. § 1206:

I.	Real Party In Interest
II	Related Appeals and Interferences
III.	Status of Claims
IV.	Status of Amendments
V.	Summary of Claimed Subject Matter
VI.	Grounds of Rejection to be Reviewed on Appeal
VII.	Argument
VIII.	Claims

IX. Evidence
X. Related Proceedings
Appendix A Claims
Appendix B Evidence
Appendix B Related Proceedings

I. REAL PARTY IN INTEREST

The real party in interest for this appeal is:

Hewlett-Packard Development Company, L.P., a Texas Limited Partnership having its principal place of business in Houston, Texas.

II. RELATED APPEALS, INTERFERENCES, AND JUDICIAL PROCEEDINGS

There are no other appeals, interferences, or judicial proceedings which will directly affect or be directly affected by or have a bearing on the Board's decision in this appeal.

III. STATUS OF CLAIMS

A. Total Number of Claims in Application

There are 20 claims pending in application.

B. Current Status of Claims

1. Claims canceled: None
2. Claims withdrawn from consideration but not canceled: None
3. Claims pending: 1-20
4. Claims allowed: None
5. Claims rejected: 1-20

C. Claims On Appeal

The claims on appeal are claims 1-20

IV. STATUS OF AMENDMENTS

No Amendment After Final Rejection has been filed with respect to the present application. Accordingly, the claims enclosed herein as Appendix A are as rejected in the Final Office Action mailed August 19, 2005.

V. SUMMARY OF CLAIMED SUBJECT MATTER

The following provides a concise explanation of the subject matter defined in each of the claims involved in the appeal, referring to the specification by page and line number and to the drawings by reference characters, as required by 37 C.F.R. § 41.37(c)(1)(v). Each element of the claims is identified by a corresponding reference to the specification and drawings where applicable. Note that the citation to passages in the specification and drawings for each claim element does not imply that the limitations from the specification and drawings should be read into the corresponding claim element.

According to one claimed embodiment, such as that of independent claim 1, a method for reducing network retrieval latency is provided. The method comprises sending a request for a data object to a server (e.g., server 102 of FIGURES 1-2, block 610 of FIGURE 6, and *see* page 3, lines 4-6 and page 6, lines 5-14). The method further comprises receiving a header portion of a response to the request (e.g., block 612 of FIGURE 6, and *see* page 3, lines 6-7 and page 6, lines 14-15), parsing the header portion for a digest value (e.g., block 614 of FIGURE 6, and *see* page 3, lines 15-16 and page 6, lines 15-16), and comparing the digest value to a digest index (e.g., digest index 318 of FIGURE 3, block 616 of FIGURE 6, and *see* page 3, lines 15-17 and page 6, lines 16-17). The method further comprises retrieving a cached data object from a cache (e.g., cache 314 of FIGURE 3) if the digest value has a match in the digest index (e.g., block 622 of FIGURE 6, and *see* page 3, lines 20-23 and page 6, lines 19-20), and sending the cached data object to a client (e.g., client 110 of FIGURES 1 and 3, block 624 of FIGURE 6, and *see* page 3, lines 20-23 and page 6, lines 20-21). If the digest value has a match in the digest index, the method further comprises informing the server to stop sending a remaining portion of the response (e.g., block 620 of FIGURE 6, and *see* page 3, lines 20-23 and page 6, lines 17-19).

According to certain embodiments, such as that of claim 4, the method further comprises receiving the remaining portion of the response from the server if no match for the

digest value is found in the digest index based on the comparing step (e.g., block 626 of FIGURE 6, and *see* page 7, lines 7-9), and sending the remaining portion of the response to the client (e.g., block 628 of FIGURE 6, and *see* page 7, lines 8-9).

According to certain embodiments, such as that of claim 5, the informing includes instructing the server to terminate a connection (*see* page 6, line 25 – page 7, line 6).

In certain embodiments, such as that of claim 19, the informing comprises: responsive to determining the digest value has a match in the digest index, performing the informing (e.g., responsive to match determination in block 618 of FIGURE 6, performing the informing of block 620 of FIGURE 6), and *see* page 6, lines 17-19).

In one claimed embodiment, such as that of independent claim 6, a method for reducing network retrieval latency comprises sending a request for a data object to a server (e.g., server 102 of FIGURES 1-2, block 610 of FIGURE 6, and *see* page 3, lines 4-6 and page 6, lines 5-14). The method further comprises receiving a server response from the server, and calculating a digest value (e.g., via digest generator 316 of FIGURE 3) for the data object based on the server response (*see* page 7, lines 14-19). The method further comprises sending a response to a client cache (e.g., cache 314 of FIGURE 3) starting with a header portion (*see* page 7, lines 16-19), the header portion including the digest value and enabling the client cache to compare the digest value to a digest index (*see* page 7, lines 19-22), retrieve a cached data object from the client cache if the digest value has a match in the digest index, and send the cached data object to a client (*see* page 7, lines 22-23). Upon receiving a message from the client cache to stop sending the response, the method further comprises stopping the sending of the response (*see* page 7, line 32 – page 8, line 2).

In one claimed embodiment, such as that of independent claim 7, a method for reducing network retrieval latency comprises receiving a first request for a data object (e.g., block 502 of FIGURE 5, and *see* page 5, line 24), and obtaining a digest value of the requested data object (e.g., block 506 of FIGURE 5, and *see* page 5, lines 26-29). The method further comprises inserting the digest value into a header portion of a response (e.g., block 506 of FIGURE 5, and *see* page 5, lines 29-32), and sending the response, starting with the header portion (e.g., block 508 of FIGURE 5, and *see* page 6, lines 1-2). Upon receiving

a second request to stop sending the response, the method further comprises stopping the sending of the response (e.g., block 510 of FIGURE 5, and *see* page 6, lines 2-4).

In one claimed embodiment, such as that of independent claim 10, a computer program product for use in conjunction with a computer system for reducing network retrieval latency is provided. The computer program product comprises logic code for sending a request for a data object to a server (e.g., server 102 of FIGURES 1-2, block 610 of FIGURE 6, and *see* page 3, lines 4-6 and page 6, lines 5-14). The computer program product further comprises logic code for receiving a header portion of a response to the request (e.g., block 612 of FIGURE 6, and *see* page 3, lines 6-7 and page 6, lines 14-15), and logic code for parsing the header portion for a digest value (e.g., block 614 of FIGURE 6, and *see* page 3, lines 15-16 and page 6, lines 15-16). The computer program product further comprises logic code for comparing the digest value to a digest index (e.g., digest index 318 of FIGURE 3, block 616 of FIGURE 6, and *see* page 3, lines 15-17 and page 6, lines 16-17), and logic code for retrieving a cached data object from a cache if the digest value has a match in the digest index (e.g., block 622 of FIGURE 6, and *see* page 3, lines 20-23 and page 6, lines 19-20). The computer program product further comprises logic code for sending the cached data object to a client (e.g., client 110 of FIGURES 1 and 3, block 624 of FIGURE 6, and *see* page 3, lines 20-23 and page 6, lines 20-21), and logic code for informing the server to stop sending a remaining portion of the response (e.g., block 620 of FIGURE 6, and *see* page 3, lines 20-23 and page 6, lines 17-19).

In certain embodiments, such as that of claim 13, the computer program product further comprises logic code for receiving the remaining portion of the response from the server if no match for the digest value is found in the digest index based on the comparing (e.g., block 626 of FIGURE 6, and *see* page 7, lines 7-9), and logic code for sending the remaining portion of the response to the client (e.g., block 628 of FIGURE 6, and *see* page 7, lines 8-9).

In certain embodiments, such as that of claim 14, the logic code for informing includes logic code for instructing the server to terminate a connection (*see* page 6, line 25 – page 7, line 6).

In certain embodiments, such as that of claim 20, the logic code for informing the server to stop sending a remaining portion of the response comprises logic code for performing the informing responsive to said logic code for comparing determining that the received digest value has a match in the digest index (e.g., responsive to match determination in block 618 of FIGURE 6, performing the informing of block 620 of FIGURE 6), and *see* page 6, lines 17-19).

In one claimed embodiment, such as that of independent claim 15, a computer program product for reducing network retrieval latency comprises logic code for sending a request for a data object to a server (e.g., server 102 of FIGURES 1-2, block 610 of FIGURE 6, and *see* page 3, lines 4-6 and page 6, lines 5-14), and logic code for receiving a server response from the server. The computer program product further comprises logic code for calculating a digest value (e.g., via digest generator 316 of FIGURE 3) for the data object based on the server response (*see* page 7, lines 14-19). The computer program product further comprises logic code for sending a response to a client cache (e.g., cache 314 of FIGURE 3) starting with a header portion (*see* page 7, lines 16-19), the header portion including the digest value and enabling the client cache to compare the digest value to a digest index (*see* page 7, lines 19-22), retrieve a cached data object from the client cache if the digest value has a match in the digest index, and send the cached data object to a client (*see* page 7, lines 22-23). The computer program product further comprises logic code for stopping the send of the response upon receiving a message from the client cache to stop sending the response (*see* page 7, line 32 – page 8, line 2).

In one claimed embodiment, such as that of independent claim 16, a computer program product for reducing network retrieval latency comprises logic code for receiving a first request for a data object (e.g., block 502 of FIGURE 5, and *see* page 5, line 24), and logic code for obtaining a digest value of the requested data object (e.g., block 506 of FIGURE 5, and *see* page 5, lines 26-29). The computer program product further comprises logic code for inserting the digest value into a header portion of a response (e.g., block 506 of FIGURE 5, and *see* page 5, lines 29-32), and logic code for sending the response, starting with the header portion (e.g., block 508 of FIGURE 5, and *see* page 6, lines 1-2). The computer program product further comprises logic code for stopping the sending of the

response upon receiving a second request to stop sending the response (e.g., block 510 of FIGURE 5, and *see* page 6, lines 2-4).

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Claims 1-20 are rejected under 35 U.S.C. § 103(a) as being unpatentable over "The HTTP Distribution and Replication Protocol," http://www.w3.org/TR/NOTE, August 25, 1997 (hereinafter "*DRP*") in view of U.S. Patent No. 5,734,898 issued to He (hereinafter "*He*").

VII. ARGUMENT

Appellant respectfully traverses the outstanding rejections of the pending claims, and requests that the Board reverse the outstanding rejections in light of the remarks contained herein. The claims do not stand or fall together. Instead, Appellant presents separate arguments for various independent and dependent claims. Each of these arguments is separately argued below and presented with separate headings and sub-heading as required by 37 C.F.R. § 41.37(c)(1)(vii).

I. Rejections Under 35 U.S.C. § 103(a) over *DRP* in view of *He*

Claims 1-20 are rejected under 35 U.S.C. § 103(a) as being unpatentable over *DRP* in view of *He*. To establish a prima facie case of obviousness, three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art cited must teach or suggest all the claim limitations. *See* M.P.E.P. §2143. Without conceding any other criteria, Appellant asserts that insufficient motivation exists for making the combination in the manner applied by the Examiner, and the applied combination of *DRP* and *He* fails to teach or suggest all elements of claims 1-20, as discussed below. Thus, for the reasons discussed below, Appellant respectfully requests that the rejection be overturned.

1. The Applied Combination Fails to Teach or Suggest All Claim Elements

Independent Claim 1 and Dependent Claims 2-3

The combination of *DRP* and *He* fails to teach or suggest all elements of independent claim 1. For instance, independent claim 1 recites:

> A method for reducing network retrieval latency, comprising the steps of:
> sending a request for a data object to a server;
> receiving a header portion of a response to said request;
> parsing said header portion for a digest value;
> comparing said digest value to a digest index;
> retrieving a cached data object from a cache if said digest value has a match in said digest index;
> sending said cached data object to a client; and
> if said digest value has a match in said digest index, informing said server to stop sending a remaining portion of said response. (Emphasis added).

The combination of *DRP* and *He* fails to teach or suggest all of the above elements of claim 1. The Final Office Action relies on *DRP* as teaching all of the above elements except for "informing said server to stop sending a remaining portion of said response", which the Office Action relies on *He* as teaching. However, *DRP* does not teach parsing a received header portion of a response for a digest value, comparing the digest value to a digest index, and retrieving a cached data object from a cache if the digest value has a match in the digest index.

Before addressing the specific elements of claim 1 further, a brief overview of *DRP* is helpful. In *DRP*, a client requests an index for a site, and the index includes content identifiers for the files of the site. Thus, the index provides a hierarchical snapshot of the set of files for a site at a particular moment in time, *see* Section 2.2 of *DRP*. The client can then download all of the files of the index. The client can further use the content identifiers to determine if any of the files in an index have the same content, and avoid downloading duplicative content. A new index can later be requested to determine if an update/modification is available for the content. That is, the client can request a new index from the site and compare the content identifiers included in the newly received index against the content identifiers of the old index to determine if any of the content has changed. If the content has changed, the client can request the updated content using a request with a

Content-ID or Differential-ID included in the header of the request. The server uses the Content-ID or Differential-ID that is included in the request to identify the appropriate content (i.e., desired version of a file) to send to the client.

In *DRP*, a comparison of two digest values can occur in any of the following contexts:

(1) a client retrieves an index file (with digests for many files) from a server: the client extracts a digest from the received index, and the client compares against the digest for client-cache entry;

(2) the client sends the digest (as Content-ID) to the server: the server compares the digest against its available content, and returns to the client content, if any, that matches; and

(3) client sends the digest (as Content-ID) via a proxy: the proxy compares the digest against cached content, and on a match returns the cache entry; otherwise, the proxy forwards the digest to the server as in context (2).

In no case of *DRP* does a server send, in response to a client request for content, a digest for the current request, and the digest comparison is then made at the client or proxy cache. In context (1) identified above, *DRP* sends an index file full of digest values, on the speculation that the client might want to retrieve content corresponding to one or more of those digest values. Such index file is not sent in response to a client request for content. Rather, the client requests the index file and uses it to identify the files of a site and the files' respective digests.

In contexts (2) and (3) identified above, *DRP* makes the digest comparison at the recipient of a REQUEST, rather than making the comparison at the recipient of a RESPONSE to a request. That is, in contexts (2) and (3), the server (or proxy) makes the digest comparison for a digest included in a request from a client. As discussed further below, the response from the server to the client may also include the digest to aid the client in confirming that the response is the desired version of a file. However, in *DRP*, the client does not use such a digest that is included in the response from the server to compare with its cache. Instead, in *DRP* a client first requests an index and uses the digests contained in the index to compare against its cache. The client makes a request for a desired file only after it determines that the desired file does not reside in its cache. Thus, while the response from the server may contain a digest to aid the client in confirming that the response is the desired version of a file, the client does not use such a received digest to compare against its cache

because the client has previously determined that such file does not reside in its cache (before requesting it from the server).

DRP does not teach or suggest a client requesting an object, receiving a response to that request where the response includes a header having a digest value, and comparing the digest value against an index to determine if the requested object is in the cache. Rather, in *DRP* the client first requests a separate index file that includes the respective content identifiers for the files of a site. The content identifiers (digests) contained in the index file may be used by the client to determine whether a desired file of the site is already possessed in the client's cache, before the client requests the file from the server. Thus, in *DRP* the client first requests an index file for a site, and then determines by comparing the digests (content identifiers) contained in the index file for the files desired by the client, whether the client already possesses such files in its cache. If the client does not possess a desired file in its cache, the client may then request from the server the file by including the respective content identifier of the desired file in its request to the server.

Claim 1 recites receiving a digest value in the header portion of a response to a request for a data object. Thus, embodiments of the present invention could be employed to alleviate requesting an index file by the client in the manner required by *DRP*. That is, *DRP* first requires a client to request an index file to obtain the digests for files of site, which is then used by the client to determine whether a desired file is possessed by its cache. Use of an embodiment encompassed by claim 1 could be used to alleviate the client first requesting an index file, wherein the client could instead request the actual content (data object) from the server, and in response to such request receive a digest value that the client can then use to determine whether the requested data object is already possessed by its cache (in which case the server is informed to stop sending the remaining portion of the data object).

DRP describes using a content-ID or differential-ID in a header. For instance, the client may use a content-ID (obtained from a previously requested index file) in a request to a server to identify a specific version of a file that is desired by the client. Further, the server may include such content-ID in its response to the client's request. However, *DRP* does not teach that the client uses these IDs for comparing against an index to determine if the requested object is in the cache. Rather, in *DRP* the client first uses the content identifiers contained in an index file to determine whether a desired file is possessed in its cache, and

thus when a file is requested from a server, the client has previously determined that it does not possess the file in its cache. Therefore, the content identifier included in a response from the server is not used by the client to compare against an index to determine if the requested file is in the client's cache (as the client has previously determined that it is not), but instead such a content identifier may be used to aid the client in verifying that the file being received from the server is the specific version desired by the client.

With regard specifically to claim 1, *DRP* does not teach or suggest all elements of claim 1 that the Final Office Action relies on *DRP* as teaching. First, *DRP* does not teach or suggest that the client compares a digest value (or the content-ID) received in a response header against an index and retrieves a cached data object from a cache if the digest value has a match in the index. Rather, using the *DRP* scheme, the client would check the content-ID against the index for a site to determine if the desired content is already possessed by the client. If the content-ID matches that of the corresponding file in the index, then the client could forego downloading this content again. In this case, the content-ID that is compared against the index for a site is not included in a header of a response to a request for an object, as no request for the object is made to the server. That is, if the content-ID matches that of the corresponding file in the index, the client foregoes requesting the file from the server in *DRP*.

Further, as mentioned above, a content-ID and a differential-ID may be used as header fields in *DRP*. However, in the case in which these IDs are used, the content-ID or Differential-ID is not compared against the index, and thus a cached data object is not retrieved from the cache if a match occurs (as no comparison is made). Rather, in the instances in which these IDs are included in a header in *DRP*, the information identified by such IDs is to be retrieved from a server (because the client has previously determined that it does not possess the corresponding content in its cache). Any comparison that is made in this instance, is made at the server to determine the corresponding content to return to the client.

In response to the above arguments, the Final Office Action maintains that *DRP* teaches the above elements of claim 1. Page 7 of the Final Office Action asserts: "In particular, the passage on the Content-Identifier Field within section 2.4 discloses that the client first sends a GET request to the server." In actuality, section 2.4 of *DRP* does not disclose that the client "first" sends a GET request to the server for a desired data object.

Rather, section 2.4 specifically teaches that a client first requests an index file that describes the structure and state of a set of data files. The first paragraph of Section 2.4 of *DRP* provides: "Given an index, it is possible for a client to determine exactly which files need to be downloaded, as well as the total size of the download." The third paragraph of Section 2.4 of *DRP* further explains:

> Note that a client can use a disk cache for data files, which is accessed using content identifiers. This means that if multiple indexes refer to a file with the same content identifier, the client can automatically detect that the file is already in the cache, and thus avoid downloading the file a second time.

Thus, the client first obtains an index file, and the content identifiers of the files listed in the index file may be used by the client to determine whether a desired file is already present in the client's cache before requesting the file a second time. That is, if the index refers to a file with the same content identifier as for a file that is in the client's cache, the client can retrieve that file from its cache and thus avoid requesting it from the server.

DRP goes on to explain in Section 2.4 thereof that since "it is possible to obtain an index for a large set of files, a mechanism is needed to obtain the correct version of each of the files that need downloading." That is, if a client determines that a desired version of a file is not possessed in its cache, then the client needs a mechanism for requesting the desired version of the file from the server. *DRP* provides that the client may use a content identifier in its request to the server for identifying the specific version of a file desired. Accordingly, *DRP* explains:

> When requesting a file, the client can include the content identifier in the HTTP GET request to the server. ... A new HTTP header field called Content-ID is used to specify the correct version of the file that is requested. The server can use the content-identifier in the Content-ID field server to determine if the requested version of the file can be delivered to the client.

Thus, this teaches that the client may include the Content-ID for a desired version of a file in a GET request to the server, and the server may use the Content-ID to identify the desired version of the file to serve to the client.

As the Final Office Action notes (on page 7 thereof), Section 2.4 of *DRP* further mentions that in responding to such a GET request: "The content identifier of the returned file should be included in the HTTP reply header using the Content-ID header field." Of

course, *DRP* does not teach that such content identifier included in the reply from the server is used by the client for comparison to a digest index such that a cached data object is retrieved from cache if the content identifier (digest value) has a match in the digest index, as in claim 1. Rather, the GET request is made by the client only after the client has determined that the desired version of the file is not contained in its cache and is therefore is needed to be downloaded from the server. While the reply from the server may include the content identifier, such content identifier is used merely to aid the client in verifying that the returned file is the version actually requested, and is not for use by the client in determining whether the requested file resides in cache (as the client has already determined that the desired file does not reside in cache in this instance). Additionally, Section 2.4 of *DRP* further explains:

> Note that the server is not required to specify a Content-ID in the reply, and that it is the responsibility of the client to verify that the reply contains the correct content identifier if a Content-ID field is present in the reply.

Again, this teaching of *DRP* clarifies that any content identifier that may be included in the response from the server is merely used by the client for verifying that the file being downloaded is the version actually requested, and is not used in anyway for determining whether the file being downloaded exists in the client's cache (as the client has previously determined that the file does not exist in its cache before issuing the GET request to download the file from the server).

In view of the above, when the teaching of *DRP* is properly considered as a whole, it clearly provides a system in which a content identifier obtained in a requested index file may first be used by a client to determine whether a desired version of a file is present in the client's cache. If the desired version is determined by the client as not present in its cache, then the client requests download of the desired version of the file (by issuing a GET request that includes the content identifier of the file desired); and the response from the server may include the content identifier, but such content identifier is used merely to aid the client in verifying that the file being downloaded is the version desired. Any such content identifier included in a response from the server is not used by the client for determining whether the file being downloaded is in the client's cache, as the client has previously determined that the file is not in its cache before requesting the file to be downloaded from the server.

Additionally, claim 1 recites "informing said server to stop sending a remaining portion of said response." *DRP* does not inform the server to stop sending a remaining portion of the response. The Final Office Action incorrectly asserts on page 8 thereof that "in order to avoid the redundant download, DRP would inherently have to cancel the request." This is simply not true. To establish inherency, the element must necessarily flow from the teaching of the *DRP* reference. The Final Office Action has failed to explain how this element necessarily flows from the teaching of the *DRP* reference, and thus has failed to establish inherency. Further, such teaching does not necessarily flow from the teaching of the *DRP* reference. Instead, as discussed above, *DRP* avoids a redundant download by a client not issuing a GET request to a server for a file that the client already possesses in its cache. Thus, a request need not be canceled, but instead the request to the server is simply not issued in the first place if the client determines that it already possesses the desired file in its cache.

The Final Office Action appears to concede that *DRP* does not teach "informing said server to stop sending a remaining portion of said response", *see* page 3 of the Final Office Action. However, the Final Office Action asserts, on page 3 thereof, that *He* teaches this element, and concludes that it would have been obvious to one of ordinary skill in the art at the time of the invention to modify *DRP* to include this teaching of *He*. However, *He* does not teach or suggest "informing said server to stop sending a remaining portion of said response." Rather, the portion of *He* relied upon by the Final Office Action refers to aborting a "transaction" (i.e., as opposed to committing the transaction), and does not refer to informing a server to stop sending a remaining portion of a response. More particularly, the Final Office Action cites to col. 3, lines 32-40 of *He*, which provides:

> FIG. 20 shows the operation of client A requesting a commit by the server of the update request described above. Client A sends commitRequest() to the server. Then the server returns commitReturn(). Similarly, for abort operation, an abort request and abort return are transferred. In this case, since the server version and client A version of object oid1 have already been updated to the same, the server will not send a version and other data to client A. This is a waste in terms of the usage of communication line.

This portion of *He* is addressing the performance of transactions and maintaining data objects consistent between a client and a server for performance of a transaction. Performing actions as "transactions" is well-known in the art, and one of ordinary skill in the art would readily appreciate that the above teaching of *He* is addressing aborting a transaction, as

opposed to committing the transaction. Aborting the transaction is not referring to informing the server to stop sending a remaining portion of a response. The above portion of *He* also mentions that the server will not send a version and other data to client A because the versions of object oid1 contained on the server and client already match. Thus, this refers to the server foregoing the sending of data to the client, rather than the client informing the server to stop sending a "remaining portion" of a response.

The Final Office Action asserts on page 9 thereof that in *He* "the abort request is sent to the server for the express purpose of stopping the transmission (i.e. stopping sending the remaining portion of the response) and thus preventing the redundant download." This is not true. The abort request in *He* is to abort a transaction, rather than commit the transaction. Contrary to the assertion by the Examiner, an abort request is not sent in *He* for the express purpose of stopping a transmission. *He* does not teach that an abort request is used to stop a transmission. Rather, as discussed further below, *He* describes a technique in which an object in a client and server cache is updated as a transaction. That is, a client can request to update an object in the cache, and can then either "commit" the transaction or "abort" the transaction. If the update is committed, the server's cache is modified to reflect the update. On the other hand, if the update is aborted, the server's cache is not modified to reflect the update, but rather the object in the server's cache remains as it was before the update. Thus, an "abort" in the transaction processing system of *He* does not address stopping a transmission at all, but instead addresses whether an update is to be committed in a server's cache or not.

He is directed to a "method and device for asynchronously updating data in a system including clients and servers each having a cache." Col. 1, lines 11-12 of *He*. *He* describes a technique for updating cached data as a transaction, where "a transaction consists of one or more operations for reading or updating objects." Col. 2, lines 21-22 of *He*. *He* describes that a Basic 2 Phase Lock (B2PL) or a Cashing 2PL (CP2L) technique may be used for updating a cached object as a transaction, *see* Col. 2, line 23 – Col. 3, line 67 of *He*. In describing the B2PL technique, *He* refers to FIGS. 16-18 in which "server cache 21 and client cache a 23 (cache of client A) contain object oid1." Col. 2, lines 28-29 of *He*. "When client A attempts to update object oid1 held in its own cache 23, it sends updateRequest(oid1, content), i.e. object identifier and contents, to the server via communication line 29." Col. 2,

lines 30-33 of *He*. "The server places a write lock on object oid1 in its cache 21, receives updated contents, and returns updateReturn(OK) for an update request." Col. 2, lines 33-35 of *He*. Thus, the client A can send updated content for modifying object oid1 to the server, and the server locks the oid1 in its cache. A lock is maintained on the server's cached object "until a commit or abort operation is performed after a read or write operation in a transaction." Col. 2, lines 60-67 of *He*.

Thereafter, the client can either request to commit the update or abort the update. If committing the update: "client A sends commitRequest() to the server. The server unlocks object oid1 in server cache 21 and returns commitReturn(oid1), i.e. object identifier, to client A." Col. 2, lines 38-41 of *He*. If aborting the update, "client A sends abortRequest() and the server returns abortReturn(oid1)." Col. 2, lines 41-42 of *He*. The object oid1 in client cache 23 is discarded when commitReturn(oid1) is received so that the integrity of the objects in each cache is maintained. *See* Col. 2, lines 42-50 of *He*. That is, if client A commits an update to object oid1 in the server's cache, the object oid1 in the client's cache (which now differs from the updated server cached object) is discarded. On the other hand, if client A aborts the update to the object oid1 in the server's cache, then the object in the server's cache is not updated and thus the object in the client's cache remains consistent with that in the server's cache.

He also teaches the C2PL technique, which "is the same as B2PL in that the server places a shared lock (read lock) or exclusive lock (write or update lock)." Col. 3, lines 5-7 of *He*. "Unlike B2PL which discards objects in the client cache between transactions, C2PL does not discard objects, but uses a log sequence number for each object at each server and client." Col. 3, lines 7-10 of *He*. Thus, in describing C2PL, *He* provides the following at col. 3, lines 32-40, which is cited by the Final Office Action:

> FIG. 20 shows the operation of client A requesting a commit by the server of the update request described above. Client A sends commitRequest() to the server. Then the server returns commitReturn(). Similarly, for abort operation, an abort request and abort return are transferred. In this case, since the server version and client A version of object oid1 have already been updated to the same, the server will not send a version and other data to client A. This is a waste in terms of the usage of communication line.

Again, this does not teach "informing said server to stop sending a remaining portion of said response" as recited by claim 1. Rather, this merely teaches that if a client commits or aborts an update to an object in a server's cache such that the object in the server's cache remains consistent with that in the client's cache, then the object in the client's cache need not be discarded and thus the server can forego sending the object to the client (because the client already possesses such object). This in no way addresses "informing said server to stop sending a remaining portion of said response" as recited by claim 1.

In view of the above, the applied combination of *DRP* and *He* fails to teach or suggest all elements of independent claim 1, and thus the rejection of claim 1 under 35 U.S.C. §103(a) should be overturned.

Claims 2-3 each depend either directly or indirectly from independent claim 1, and thus claims 2-3 each inherit all elements of claim 1. Therefore, claims 2-3 are each allowable over the applied combination of *DRP* and *He* at least for the reasons discussed above with claim 1. As such, Appellant respectfully requests that the rejection of claims 2-3 be overturned.

Dependent Claim 4

Dependent claim 4 depends from claim 1 and thus inherits all elements of claim 1. Accordingly, claim 4 is allowable over the applied combination of *DRP* and *He* at least for the reasons discussed above with claim 1. Additionally, claim 4 further recites "receiving said remaining portion of said response from said server if no match for said digest value is found in said digest index based on said comparing step" (emphasis added). The combination of *DRP* and *He* further fails to teach or suggest this element of claim 4. The Final Office Action asserts that *DRP* teaches this element. However, as discussed above with claim 1, *DRP* does not teach performing the comparing step for a digest value received in a response. Rather, in *DRP* the client compares digest values contained in a received index file with its cache, and the client requests a file from a server only after the client determines that the file is not present in the client's cache. Thus, Appellant respectfully requests that the rejection of claim 4 be overturned for this further reason.

Dependent Claim 5

Dependent claim 5 depends from claim 1 and thus inherits all elements of claim 1. Accordingly, claim 5 is allowable over the applied combination of *DRP* and *He* at least for the reasons discussed above with claim 1. Additionally, claim 5 further recites "instructing said server to terminate a connection." The combination of *DRP* and *He* further fails to teach or suggest this element of claim 5. The Final Office Action fails to identify any teaching in *DRP* or *He* in which a server is instructed to terminate a connection. As discussed above with claim 1, neither *DRP* nor *He* teaches or suggests informing a server to stop sending a remaining portion of a response. Further, neither reference teaches or suggests instructing the server to terminate a connection. Thus, Appellant respectfully requests that the rejection of claim 5 be overturned for this further reason.

Dependent Claim 19

Dependent claim 19 depends from claim 1 and thus inherits all elements of claim 1. Accordingly, claim 19 is allowable over the applied combination of *DRP* and *He* at least for the reasons discussed above with claim 1. Additionally, claim 19 further recites "wherein said informing comprises: responsive to determining said digest value has a match in said digest index, performing said informing." The combination of *DRP* and *He* further fails to teach or suggest this element of claim 19. As discussed above with claim 1, neither *DRP* nor *He* teaches or suggests informing a server to stop sending a remaining portion of a response. Moreover, neither reference teaches or suggests performing such informing "responsive to determining said digest value has a match in said digest index". Thus, Appellant respectfully requests that the rejection of claim 19 be overturned for this further reason.

Independent Claim 6

Independent claim 6 recites:

> A method for reducing network retrieval latency, comprising the steps of:
> sending a request for a data object to a server;
> receiving a server response from said server;
> calculating a digest value for said data object based on said server response;
> sending a response to a client cache starting with a header portion,

said header portion including said digest value and enabling said client cache to compare said digest value to a digest index, retrieve a cached data object from said client cache if said digest value has a match in said digest index, and send said cached data object to a client; and

upon receiving a message from said client cache to stop sending said response, stopping the sending of said response. (Emphasis added).

The combination of *DRP* and *He* fails to teach or suggest all of the above elements of claim 6. The Final Office Action relies on *DRP* as teaching all of the elements of claim 6 except for "upon receiving a message from said client cache to stop sending said response, stopping the sending of said response", which the Office Action relies on *He* as teaching. However, *DRP* does not teach "sending a response to a client cache starting with a header portion, said header portion including said digest value and enabling said client cache to compare said digest value to a digest index, retrieve a cached data object from said client cache if said digest value has a match in said digest index, and send said cached data object to a client" (emphasis added).

As discussed above with claim 1, in *DRP* a server sends the client an index file containing digest values for various files available at a site provided by the server. *DRP* does not teach or suggest including a digest value in a header portion of a response to a client where a cached data object is retrieved from the client cache if the digest value has a match in the digest index. As discussed above with claim 1, in *DRP* a client first determines whether an object is in its cache before making a request to the server. Thus, even if the response from the server includes a content identifier, such content identifier is not then used to retrieve a cached data object if such content identifier (digest) has a match in a digest index. Accordingly, *DRP* fails to teach or suggest at least this element of claim 6.

Further, neither *DRP* nor *He* teaches or suggests "upon receiving a message from said client cache to stop sending said response, stopping the sending of said response". As discussed above with claim 1, *DRP* neither expressly nor inherently teaches receiving from a client a message to stop sending a response. As also discussed above with claim 1, *He* does not teach or suggest stopping the sending of a response either. Thus, the combination of *DRP* and *He* fails to teach or suggest this further element of claim 6.

In view of the above, claim 6 is not obvious under 35 U.S.C. §103(a) because the applied combination of *DRP* and *He* fails to teach or suggest all elements of claim 6. Therefore, Appellant respectfully requests that the rejection of claim 6 be overturned.

Independent Claim 7 and Dependent Claims 8-9

Independent claim 7 recites:

> A method for reducing network retrieval latency, comprising the steps of:
> receiving a first request for a data object;
> obtaining a digest value of said requested data object;
> inserting said digest value into a header portion of a response;
> sending said response, starting with said header portion; and
> upon receiving a second request to stop sending said response, stopping the sending of said response. (Emphasis added).

The combination of *DRP* and *He* fails to teach or suggest all of the above elements of claim 7. The Final Office Action relies on *DRP* as teaching all of the elements of claim 7 except for "upon receiving a second request to stop sending said response, stopping the sending of said response", which the Office Action relies on *He* as teaching. However, neither *DRP* nor *He* teaches or suggests "upon receiving a second request to stop sending said response, stopping the sending of said response". As discussed above with claim 1, *DRP* neither expressly nor inherently teaches receiving from a client a request to stop sending a response. As also discussed above with claim 1, *He* does not teach or suggest stopping the sending of a response either.

In view of the above, claim 7 is not obvious under 35 U.S.C. §103(a) because the applied combination of *DRP* and *He* fails to teach or suggest all elements of claim 7. Therefore, Appellant respectfully requests that the rejection of claim 7 be overturned.

Claims 8-9 each depend either directly or indirectly from independent claim 7, and thus claims 8-9 each inherit all elements of claim 7. Therefore, claims 8-9 are each allowable over the applied combination of *DRP* and *He* at least for the reasons discussed above with claim 7. As such, Appellant respectfully requests that the rejection of claims 8-9 be overturned.

Independent Claim 10 and Dependent Claims 11-12

Independent claim 10 recites:

> A computer program product for use in conjunction with a computer system for reducing network retrieval latency, comprising:
> logic code for sending a request for a data object to a server;
> logic code for receiving a header portion of a response to said request;
> logic code for parsing said header portion for a digest value;
> logic code for comparing said digest value to a digest index;
> logic code for retrieving a cached data object from a cache if said digest value has a match in said digest index;
> logic code for sending said cached data object to a client; and
> logic code for informing said server to stop sending a remaining portion of said response.

The combination of *DRP* and *He* fails to teach or suggest all of the above elements of claim 10. The Final Office Action relies on *DRP* as teaching all of the elements except for "logic code for informing said server to stop sending a remaining portion of said response", which the Office Action relies on *He* as teaching. However, *DRP* does not teach "logic code for receiving a header portion of a response to said request; ...parsing said header portion for a digest value; ... [and] retrieving a cached data object from a cache if said digest value has a match in said digest index".

As discussed above with claim 1, in *DRP* a server sends the client an index file containing digest values for various files available at a site provided by the server. *DRP* does not teach or suggest including a digest value in a header portion of a response to a client where a cached data object is retrieved from cache if the digest value has a match in the digest index. As discussed above with claim 1, in *DRP* a client first determines whether an object is in its cache before making a request to the server. Thus, even if the response from the server includes a content identifier, such content identifier is not then used to compare with a digest index and retrieve a cached data object if such content identifier (digest) has a match in the digest index. Accordingly, *DRP* fails to teach or suggest at least this element of claim 10.

Further, neither *DRP* nor *He* teaches or suggests "logic code for informing said server to stop sending a remaining portion of said response". As discussed above with claim 1, *DRP* neither expressly nor inherently teaches a client informing said server to stop sending

a remaining portion of a response. As also discussed above with claim 1, *He* does not teach or suggest informing a server to stop sending a remaining portion of a response either.

In view of the above, claim 10 is not obvious under 35 U.S.C. §103(a) because the applied combination of *DRP* and *He* fails to teach or suggest all elements of claim 10. Therefore, Appellant respectfully requests that this rejection be overturned.

Claims 11-12 each depend either directly or indirectly from independent claim 10, and thus claims 11-12 each inherit all elements of claim 10. Therefore, claims 11-12 are each allowable over the applied combination of *DRP* and *He* at least for the reasons discussed above with claim 10. As such, Appellant respectfully requests that the rejection of claims 11-12 be overturned.

Dependent Claim 13

Dependent claim 13 depends from claim 10 and thus inherits all elements of claim 10. Accordingly, claim 13 is allowable over the applied combination of *DRP* and *He* at least for the reasons discussed above with claim 10. Additionally, claim 13 further recites "logic code for receiving said remaining portion of said response from said server if no match for said digest value is found in said digest index based on said comparing" (emphasis added). The combination of *DRP* and *He* further fails to teach or suggest this element of claim 13. The Final Office Action asserts that *DRP* teaches this element. However, as discussed above with claim 10, *DRP* does not teach comparing a digest value received in a response. Rather, in *DRP* the client compares digest values contained in a received index file with its cache, and the client requests a file from a server only after the client determines that the file is not present in the client's cache. Thus, Appellant respectfully requests that the rejection of claim 13 be overturned for this further reason.

Dependent Claim 14

Dependent claim 14 depends from claim 10 and thus inherits all elements of claim 10. Accordingly, claim 14 is allowable over the applied combination of *DRP* and *He* at least for the reasons discussed above with claim 10. Additionally, claim 14 further recites "logic code for instructing said server to terminate a connection." The combination of *DRP* and *He* further fails to teach or suggest this element of claim 14. The Final Office Action fails to

identify any teaching in *DRP* or *He* in which a server is instructed to terminate a connection. As discussed above with claim 10, neither *DRP* nor *He* teaches or suggests informing a server to stop sending a remaining portion of a response. Further, neither reference teaches or suggests instructing the server to terminate a connection. Thus, Appellant respectfully requests that the rejection of claim 14 be overturned for this further reason.

Dependent Claim 20

Dependent claim 20 depends from claim 10 and thus inherits all elements of claim 10. Accordingly, claim 20 is allowable over the applied combination of *DRP* and *He* at least for the reasons discussed above with claim 10. Additionally, claim 20 further recites "logic code for performing said informing responsive to said logic code for comparing determining that said received digest value has a match in said digest index." The combination of *DRP* and *He* further fails to teach or suggest this element of claim 20. As discussed above with claim 10, neither *DRP* nor *He* teaches or suggests informing a server to stop sending a remaining portion of a response. Moreover, neither reference teaches or suggests performing such informing "responsive to said logic code for comparing determining that said received digest value has a match in said digest index". Thus, Appellant respectfully requests that the rejection of claim 20 be overturned for this further reason.

Independent Claim 15

Independent claim 15 recites:

> A computer program product for reducing network retrieval latency, comprising:
> logic code for sending a request for a data object to a server;
> logic code for receiving a server response from said server;
> logic code for calculating a digest value for said data object based on said server response;
> logic code for sending a response to a client cache starting with a header portion, said header portion including said digest value and enabling said client cache to compare said digest value to a digest index, retrieve a cached data object from said client cache if said digest value has a match in said digest index, and send said cached data object to a client; and
> logic code for stopping the send of said response upon receiving a message from said client cache to stop sending said response. (Emphasis added).

The combination of *DRP* and *He* fails to teach or suggest all of the above elements of claim 15. The Final Office Action relies on *DRP* as teaching all of the elements except for "logic code for stopping the send of said response upon receiving a message from said client cache to stop sending said response", which the Office Action relies on *He* as teaching. However, *DRP* does not teach "logic code for sending a response to a client cache starting with a header portion, said header portion including said digest value and enabling said client cache to compare said digest value to a digest index, retrieve a cached data object from said client cache if said digest value has a match in said digest index, and send said cached data object to a client".

As discussed above with claim 1, in *DRP* a server sends the client an index file containing digest values for various files available at a site provided by the server. *DRP* does not teach or suggest including a digest value in a header portion of a response to a client where the client compares the digest value to a digest index, and retrieve a cached data object from the client cache if the digest value has a match in the digest index. As discussed above with claim 1, in *DRP* a client first determines whether an object is in its cache before making a request to the server. Thus, even if the response from the server includes a content identifier, such content identifier is not then used to compare with a digest index and retrieve a cached data object if such content identifier (digest) has a match in the digest index. Accordingly, *DRP* fails to teach or suggest at least this element of claim 15.

Further, neither *DRP* nor *He* teaches or suggests "logic code for stopping the send of said response upon receiving a message from said client cache to stop sending said response". As discussed above with claim 1, *DRP* neither expressly nor inherently teaches receiving from a client a message to stop sending a response. As also discussed above with claim 1, *He* also fails to teach or suggest stopping the send of a response upon receiving a message from the client cache to stop sending the response.

In view of the above, claim 15 is not obvious under 35 U.S.C. §103(a) because the applied combination of *DRP* and *He* fails to teach or suggest all elements of claim 15. Therefore, Appellant respectfully requests that the rejection of claim 15 be overturned.

Independent Claim 16 and Dependent Claims 17-18

Independent claim 16 recites:

> A computer program product for reducing network retrieval latency, comprising:
> logic code for receiving a first request for a data object;
> logic code for obtaining a digest value of said requested data object;
> logic code for inserting said digest value into a header portion of a response;
> logic code for sending said response, starting with said header portion; and
> logic code for stopping the sending of said response upon receiving a second request to stop sending said response. (Emphasis added).

The combination of *DRP* and *He* fails to teach or suggest all of the above elements of claim 16. The Final Office Action relies on *DRP* as teaching all of the elements except for "logic code for stopping the sending of said response upon receiving a second request to stop sending said response", which the Office Action relies on *He* as teaching. However, neither *DRP* nor *He* teaches or suggests "logic code for stopping the sending of said response upon receiving a second request to stop sending said response". As discussed above with claim 1, *DRP* neither expressly nor inherently teaches receiving from a client a request to stop sending a response. As also discussed above with claim 1, *He* does not teach or suggest this element either.

In view of the above, claim 16 is not obvious under 35 U.S.C. §103(a) because the applied combination of *DRP* and *He* fails to teach or suggest all elements of claim 16. Therefore, Appellant respectfully requests that the rejection of claim 16 be overturned.

Claims 17-18 each depend either directly or indirectly from independent claim 16, and thus claims 17-18 each inherit all elements of claim 16. Therefore, claims 17-18 are each allowable over the applied combination of *DRP* and *He* at least for the reasons discussed above with claim 16. As such, Appellant respectfully requests that the rejection of claims 17-18 be overturned.

2. Insufficient Motivation to Combine DRP and He as Applied

Additionally, the rejection of claims 1-20 should be overturned because the Final Office Action fails to establish proper suggestion or motivation to combine the teachings of *DRP* and *He*. The Final Office Action asserts at page 3 thereof: "It would have been obvious to one of ordinary skill in the art at the time of the invention to modify DRP by informing the server to stop sending a remaining portion of said response for the purpose of preventing the download of a file already stored in the cache, as taught by He." The language of the recited motivation is circular in nature, stating that it is obvious to make the modification because it is obvious to achieve the result. That is, such language is merely a statement that the *DRP* reference can be modified to inform the server to stop sending a remaining portion of a response so that the remaining portion of the response will not be sent. This does not identify any motivation for modifying *DRP* in this manner. As described above, in *DRP* a client does not request a file until after it has first determined that such file is not possessed by the client's cache. Thus, *DRP* provides no motivation for stopping the sending of a remaining portion of a response. Indeed, modifying *DRP* in this manner would appear to modify a principle of its operation because the client has previously determined that it needs the file being downloaded in *DRP*. Additionally, as discussed above, *He* provides no teaching or suggestion of informing a server to stop sending a response, and thus *He* provides no motivation for making such a drastic change to the operation of *DRP*.

It is well settled that the mere fact that references can be combined or modified does not render the resultant combination or modification obvious unless the prior art also suggests the desirability of the combination or modification. *In re Mills*, 916 F.2d 680, 16 U.S.P.Q.2d 1430 (Fed. Cir. 1990), as cited in M.P.E.P. § 2143.01. As detailed above, no such motivation has been identified here, and thus Appellant requests that the rejection of claims 1-20 be overturned for this further reason.

VIII. CLAIMS

A copy of the claims involved in the present appeal is attached hereto as Appendix A. As indicated above, the claims in Appendix A are as rejected in the Final Office Action mailed August 19, 2005.

IX. EVIDENCE

As noted in Appendix B hereto, no evidence pursuant to §§ 1.130, 1.131, or 1.132 or entered by or relied upon by the examiner is being submitted.

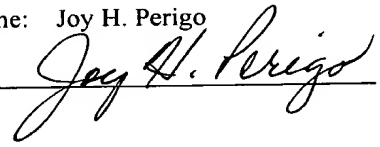
X. RELATED PROCEEDINGS

As noted in Appendix C hereto, no related proceedings are referenced in II. above, or copies of decisions in related proceedings are not provided.

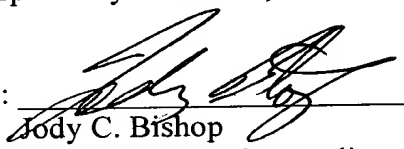
I hereby certify that this correspondence is being deposited with the United States Postal Service as Express Mail, Label No. EV 568257431US in an envelope addressed to: M/S Appeal Brief-Patents, Commissioner for Patents, Alexandria, VA 22313.

Date of Deposit: January 20, 2006

Typed Name: Joy H. Perigo

Signature: 

Respectfully submitted,

By: 

Jody C. Bishop
Attorney/Agent for Applicant(s)
Reg. No. 44,034
Date: January 20, 2006
Telephone No. (214) 855-8007

APPENDIX A

Claims Involved in the Appeal of Application Serial No. 09/825,661

1. A method for reducing network retrieval latency, comprising the steps of:

sending a request for a data object to a server;

receiving a header portion of a response to said request;

parsing said header portion for a digest value;

comparing said digest value to a digest index;

retrieving a cached data object from a cache if said digest value has a match in said digest index;

sending said cached data object to a client; and

informing said server to stop sending a remaining portion of said response.

2. The method of claim 1, further comprising the steps of:

checking said cache for said data object before sending said request to said server; and

sending said data object to said client if said data object is found in said cache.

3. The method of claim 1, wherein said digest index is a hash table.

4. The method of claim 1, further comprising the steps of:

receiving said remaining portion of said response from said server if no match for said digest value is found in said digest index based on said comparing step; and

sending said remaining portion of said response to said client.

5. The method of claim 1, wherein said informing includes the step of:

instructing said server to terminate a connection.

6. A method for reducing network retrieval latency, comprising the steps of:

sending a request for a data object to a server;

receiving a server response from said server;

calculating a digest value for said data object based on said server response;

sending a response to a client cache starting with a header portion, said header portion including said digest value and enabling said client cache to compare said digest value to a digest index, retrieve a cached data object from said client cache if said digest value has a match in said digest index, and send said cached data object to a client; and

upon receiving a message from said client cache to stop sending said response, stopping the sending of said response.

7. A method for reducing network retrieval latency, comprising the steps of:

receiving a first request for a data object;

obtaining a digest value of said requested data object;

inserting said digest value into a header portion of a response;

sending said response, starting with said header portion; and

upon receiving a second request to stop sending said response, stopping the sending of said response.

8. The method of claim 7, wherein said obtaining includes the step of:

retrieving said digest value from a hash table.

9. The method of claim 7, wherein said obtaining includes the step of:

calculating said digest value based on contents of said data object.

10. A computer program product for use in conjunction with a computer system for reducing network retrieval latency, comprising:

logic code for sending a request for a data object to a server;

logic code for receiving a header portion of a response to said request;

logic code for parsing said header portion for a digest value;

logic code for comparing said digest value to a digest index;

logic code for retrieving a cached data object from a cache if said digest value has a match in said digest index;

logic code for sending said cached data object to a client; and

logic code for informing said server to stop sending a remaining portion of said response.

11. The computer program product of claim 10, further comprising:

logic code for checking said cache for said data object before sending said request to said server; and

logic code for sending said data object to said client if said data object is found in said cache.

12. The computer program product of claim 10, wherein said digest index is a hash table.

13. The computer program product of claim 10, further comprising:

logic code for receiving said remaining portion of said response from said server if no match for said digest value is found in said digest index based on said comparing; and

logic code for sending said remaining portion of said response to said client.

14. The computer program product of claim 10, wherein said logic code for informing includes:

logic code for instructing said server to terminate a connection.

15. A computer program product for reducing network retrieval latency, comprising:

logic code for sending a request for a data object to a server;

logic code for receiving a server response from said server;

logic code for calculating a digest value for said data object based on said server response;

logic code for sending a response to a client cache starting with a header portion, said header portion including said digest value and enabling said client cache to compare said digest value to a digest index, retrieve a cached data object from said client cache if said digest value has a match in said digest index, and send said cached data object to a client; and

logic code for stopping the send of said response upon receiving a message from said client cache to stop sending said response.

16. A computer program product for reducing network retrieval latency, comprising:

logic code for receiving a first request for a data object;

logic code for obtaining a digest value of said requested data object;

logic code for inserting said digest value into a header portion of a response;

logic code for sending said response, starting with said header portion; and

logic code for stopping the sending of said response upon receiving a second request to stop sending said response.

17. The computer program product of claim 16, wherein said logic code for obtaining includes:

logic code for retrieving said digest value from a hash table.

18. The computer program product of claim 16, wherein said logic code for obtaining includes:

logic code for calculating said digest value based on contents of said data object.

19. The method of claim 1 wherein said informing comprises:

responsive to determining said digest value has a match in said digest index, performing said informing.

20. The computer program product of claim 10, wherein said logic code for informing said server to stop sending a remaining portion of said response comprises:

logic code for performing said informing responsive to said logic code for comparing determining that said received digest value has a match in said digest index.

APPENDIX B

Evidence

None.

APPENDIX C

Related Proceedings

None.